Reproducible Clinical Reporting Systems: A Workflow-Oriented Extension of TLFQC for Versioned, Traceable, and Codeless Biostatistics Outputs

Claude M. Robles
Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS, USA.
claudemrobles@ku.edu

Abstract

The reliability of clinical reporting in biostatistics is increasingly challenged by the complexity of multi-study data integration, regulatory scrutiny, and the need for transparent, reproducible outputs. Traditional approaches rely on manual scripting and ad hoc validation, which introduce fragility and inconsistency across organizational boundaries. This paper proposes a workflow-oriented extension of the TLFQC framework, originally designed for automated and codeless generation of tables, listings, and figures. The extension emphasizes versioning, traceability, and governance as integral components of a reproducible clinical reporting system. By embedding TLFQC within a continuous integration and delivery pipeline, the proposed architecture enables systematic tracking of data provenance, parameter changes, and output evolution across study milestones. The system supports both regulatory submission requirements and internal audit readiness through immutable audit trails and role-based access controls. A codeless interface lowers the barrier for biostatisticians and clinical programmers, reducing reliance on ad hoc R or SAS scripts while maintaining flexibility for complex analytical workflows. Structural trade-offs between flexibility and reproducibility are examined, including the tension between GUI-driven specification and version control of analysis definitions. The paper further explores deployment sustainability, robustness under varying data formats, and fairness implications related to standardization of output representation across diverse therapeutic areas. Policy and governance considerations are discussed in the context of 21st Century Cures Act and FDA guidance on computerized systems. The proposed extension positions TLFQC not merely as a point solution but as a foundational layer for reproducible, auditable, and interoperable clinical reporting infrastructure.

Keywords

clinical reporting, reproducibility, TLFQC, version control, traceability, codeless biostatistics, workflow automation, regulatory compliance, audit trail, FAIR principles.

1. Introduction

The production of tables, listings, and figures constitutes a cornerstone of clinical trial reporting, serving as the primary medium through which statistical findings are communicated to regulators, investigators, and sponsors. Despite advances in statistical software and data management platforms, the generation of TLFs remains a labor-intensive process fraught with opportunities for error, inconsistency, and opaque decision-making. Recent high-profile retractions and regulatory findings have underscored the urgency of

embedding reproducibility into the fabric of clinical reporting workflows. The adoption of principles such as FAIR (Findable, Accessible, Interoperable, Reusable) has gained traction, yet their operationalization in the context of biostatistical outputs remains uneven.

The TLFQC platform, as described in recent conference proceedings [18], represents a significant step toward automating TLF generation and validation in a codeless environment. By leveraging R Shiny as an underlying engine, TLFQC allows users to specify output requirements through a graphical interface, thereby abstracting away the underlying programming details. However, the original framework is primarily oriented toward single-point generation and validation rather than the end-to-end management of versioned, traceable outputs across the lifecycle of a clinical study. This paper argues that a workflow-oriented extension is necessary to address the systemic challenges of reproducibility at scale.

A workflow-oriented approach treats TLF generation not as an isolated task but as a sequence of connected steps involving data extraction, specification, execution, validation, review, and archival. Each step must be recorded with sufficient metadata to enable reconstruction of any output from its inputs and parameters. The extended system proposed here integrates TLFQC with version control repositories, continuous integration pipelines, and audit log databases to create a persistent, immutable record of every output state. Such an infrastructure supports not only regulatory compliance but also internal quality improvement and cross-study meta-analyses.

The remainder of this paper is organized as follows. Section 2 reviews the background and motivations for reproducible clinical reporting. Section 3 details the system architecture that extends TLFQC. Section 4 discusses workflow integration and governance. Section 5 focuses on versioning and traceability mechanisms. Section 6 examines the codeless paradigm and its implications for validation. Section 7 addresses deployment and sustainability considerations. Section 8 explores fairness and policy implications. Section 9 concludes with a synthesis and future research directions.

2. Background and Motivation

Reproducibility in computational science has been extensively discussed in the literature, with landmark contributions highlighting the replication crisis across disciplines [1]. In clinical biostatistics, the stakes are elevated by regulatory reliance on submitted TLFs for drug approval decisions. The U.S. Food and Drug Administration has issued guidance on electronic submissions and computerized systems that implicitly require traceability and audit trails [2][3]. Yet many organizations still rely on informal versioning practices, such as saving script files with date-stamped names, which are fragile and difficult to audit.

The TLFQC framework [18] addresses part of the problem by automating output generation and providing a codeless interface that reduces manual programming errors. However, it does not inherently manage the provenance of input datasets or the configuration parameters that change over the course of a study. As studies evolve through protocol amendments, data cleaning cycles, and midcourse analyses, the set of TLF outputs must be updated and revalidated. Without systematic version control, each update risks introducing discrepancies that are hard to detect.

Several initiatives have attempted to bring reproducibility principles into clinical reporting. The concept of executable research compendia, where analysis code and data are bundled with a complete computational environment, has been advocated for statistical analyses [4][5]. In the pharmaceutical industry, standard data tabulation models such as SDTM and ADaM

impose structure on input data but do not prescribe how outputs should be versioned. The FAIR principles [6] emphasize machine-actionability and persistence of metadata, which can be applied to TLF outputs as digital objects with unique identifiers.

The motivation for a workflow-oriented extension of TLFQC arises from the recognition that reproducibility is a property of the entire system, not of a single tool. A TLF generated today may need to be reproduced months later under a different version of the underlying software or with a corrected dataset. The system must therefore capture not only the final output but the entire chain of transformations. This requirement aligns with the concept of provenance in scientific workflows [7]. By embedding TLFQC within a larger infrastructure, we can ensure that every output is anchored to a specific commit in the version control repository, a specific environment snapshot, and a specific set of user credentials.

3. System Architecture

The proposed architecture extends TLFQC by introducing three layers: a specification layer, an execution layer, and an archival layer. The specification layer, built upon the TLFQC graphical user interface, allows users to define output templates, linking them to source datasets and analysis variables. Rather than storing these specifications only in memory or in flat files, the system serializes them into a structured format (e.g., JSON or YAML) that is committed to a version control repository at the time of creation. Each specification commit receives a unique hash, forming the basis for traceability.

The execution layer is decoupled from the user interface via a queue-based architecture. When a user requests TLF generation, the specification commit hash is passed to a job scheduler that provisions a containerized environment with a known version of R, Shiny, and TLFQC. The container also includes a read-only snapshot of the relevant datasets, whose version is recorded in a metadata database. This approach ensures that the same specification combined with the same data version always produces identical outputs, regardless of when or where it is executed [8]. The execution layer also performs automated validation checks: for example, verifying that the number of records in a listing matches the count in the source dataset or that output formatting adheres to predefined style guides.

The archival layer stores each generated output file along with its provenance record. Provenance includes the specification commit hash, dataset commit hash, container image identifier, execution timestamp, and the identity of the requesting user. These records are written to an immutable append-only database, such as a blockchain-inspired ledger or a write-once-read-many store. Audit logs are accessible to reviewers and regulators through a dedicated interface that enforces role-based access controls. By maintaining a permanent history of output states, the system supports forensic reproducibility: any output can be retroactively linked to its exact inputs and parameters [9].

A critical architectural trade-off arises between flexibility and reproducibility. The codeless interface offers ease of use but may obscure the precise mapping from user actions to specification commits. For example, if a user modifies a template parameter through a drag-and-drop widget, the system must detect that change and create a new specification commit before execution. If the change is made transiently and not committed, the output becomes untraceable. The architecture enforces the rule that all execution must reference a committed specification, preventing ad hoc modifications. This rigidity may frustrate users accustomed to iterative trial-and-error, but it is essential for maintaining an auditable chain of custody [10].

4. Workflow Integration and Governance

Integrating the extended TLFQC into existing clinical workflows requires careful design of governance policies. In a typical clinical study, TLF generation is part of a larger process that includes statistical analysis plan development, data management, and medical writing. The system must interface with other enterprise tools, such as electronic data capture systems, clinical data repositories, and electronic document management systems. Interoperability is achieved through standard application programming interfaces that exchange metadata about study milestones and locked datasets.

Governance is enforced through a formal sign-off process. Before TLF outputs are used for submission or decision-making, they must pass through a review cycle that involves biostatisticians, clinical programmers, and medical reviewers. The workflow management module of the proposed extension routes outputs through predefined approval chains, with each reviewer's action recorded in the audit log. If a reviewer requests a change to the specification, the system automatically creates a new version and re-executes the outputs, initiating a new review cycle. This iterative refinement is transparent and fully documented, satisfying both internal standard operating procedures and regulatory expectations.

One of the challenges in governance is managing the tension between change control and agility. Clinical studies often face tight timelines, and requiring a full commit-and-review cycle for every minor formatting tweak can introduce delays. The system mitigates this by supporting draft and frozen states. In draft mode, users can experiment with specifications without committing changes to the immutable audit trail; only when a specification is frozen does it become subject to full version control and review. This dual-mode approach balances exploratory flexibility with audit integrity.

Cross-study governance is another dimension. Large pharmaceutical companies may have hundreds of studies running concurrently, each with its own TLF specifications. The extended TLFQC architecture supports a library of reusable templates that are governed centrally. Changes to a master template are automatically propagated to all studies that reference it, but only after a rigorous validation process. Such centralized governance reduces redundancy and ensures consistency across therapeutic areas, which is increasingly important for meta-analyses and integrated summaries of safety and efficacy [11].

5. Versioning and Traceability

Versioning in the extended TLFQC system operates at multiple granularities. At the coarsest level, each study milestone (e.g., database lock, interim analysis, final analysis) corresponds to a tagged release of the entire TLF output set. At a finer level, individual specification files and dataset snapshots are versioned independently, enabling precise reconstruction of any output. The system leverages Git as the underlying version control engine, but it abstracts away the complexity of branch management and merging from end users. Instead, users interact with a dashboard that displays a timeline of specification versions, with visual indicators for changes.

Traceability goes beyond version labels. The audit trail for each output includes a digital fingerprint (hash) of the output file itself, allowing integrity checks against tampering. If an output file is later found to be corrupted or manually altered, its hash will not match the recorded value, triggering an alert. This cryptographic linking is crucial for regulatory compliance, as it provides non-repudiation of the output state at the time of generation [12].

The traceability framework also supports reproducibility across toolchains. Because the execution environment is containerized, the system can record the exact version of every R

package used, including transitive dependencies. This metadata is stored in a dependency manifest that accompanies the output. If a future analyst wishes to reproduce the output using a different environment, the manifest provides the necessary information to replicate the original software stack. In cases where package updates break compatibility, the system can access archived container images from a registry, ensuring that reproduction remains possible even as the external ecosystem evolves [13].

6. Codeless Paradigm and Validation

The codeless paradigm of TLFQC lowers the technical barrier for biostatisticians who may not be proficient in R or SAS programming. However, the elimination of code does not eliminate the need for validation. In traditional programming approaches, validation often involves reviewing the script logic line by line. In a codeless system, validation must shift to verifying the specification's semantics and the correctness of the generated output. The proposed extension incorporates formal specification checking: the system parses user-defined templates and cross-references them against a domain ontology of statistical reporting elements. For example, if a user requests a Kaplan-Meier plot but fails to specify the time-to-event variables, the system flags the omission before execution.

Validation also includes regression testing. When a specification is updated, the system automatically re-executes all previously generated outputs that depend on the changed configuration and compares them with the archived versions. Differences are highlighted in a diff viewer, allowing reviewers to quickly assess whether the change is benign or introduces unintended modifications. This regression testing capability is critical for maintaining consistency across study updates and reduces the manual effort of revalidation [14].

The codeless interface, combined with automated validation, can reduce the incidence of common errors such as mismatched variable names, incorrect sorting orders, or inconsistent footnote formatting. However, it also introduces a dependency on the graphical interface's correctness and usability. If the interface misinterprets a user's intention, the resulting output may be erroneous but appear correct. To mitigate this risk, the system supports peer review of specifications through a commenting and approval workflow, where multiple experts examine the specification definitions before execution [15].

7. Deployment and Sustainability

Deploying the extended TLFQC system within an organization requires considerations of scalability, security, and maintainability. The architecture is designed to run on a combination of on-premises and cloud infrastructure, with sensitive patient data remaining behind organizational firewalls while container images and templates can be shared via private repositories. Horizontal scalability is achieved by pooling compute resources for TLF generation, which can be parallelized across studies.

Sustainability over the long term demands that the system be maintained as software dependencies evolve. TLFQC itself, being an R Shiny application, is subject to changes in the R ecosystem, package versions, and Shiny server updates. The containerization strategy insulates the execution environment from host system updates, but the container images themselves must be periodically rebuilt to incorporate security patches. The proposed extension includes a continuous integration pipeline that automatically rebuilds and tests container images on a regular schedule, flagging any failures that result from upstream changes [16].

Another aspect of sustainability is the preservation of output archives. As studies conclude, the audit trail and output files must be retained for regulatory periods that can exceed fifteen years. The system automatically migrates archival records to long-term storage formats (e.g., PDF/A for documents, Parquet for data files) and verifies checksums periodically to detect bit rot. This proactive preservation strategy ensures that outputs remain accessible and verifiable decades after generation [17].

8. Fairness and Policy Implications

The standardization of TLF outputs through a codeless platform raises questions about fairness and equity in clinical reporting. Standardization can improve consistency across studies and reduce the risk of selective reporting, but it may also impose a one-size-fits-all approach that inadequately captures the nuances of certain therapeutic areas. For example, outputs for rare diseases may require custom visualizations that are not supported by generic templates. The extended TLFQC system addresses this by allowing user-defined extensions through a plugin architecture, but these extensions themselves must be versioned and validated, creating a tension between flexibility and reproducibility.

Policy implications extend to the regulatory landscape. The 21st Century Cures Act and the FDA's guidance on electronic source data emphasize the need for audit trails and traceability in clinical systems [19]. The proposed architecture aligns with these requirements by providing a complete chain of custody for TLF outputs. However, regulatory acceptance of codeless systems is still evolving. Regulators may require evidence that the graphical interface does not introduce unknown biases or errors. The system's built-in validation checks and audit trails can serve as evidence for regulatory filings, but organizations must be prepared to justify their reliance on such tools during inspections.

From a fairness perspective, the transition to codeless systems could reduce the cognitive load on junior biostatisticians and clinical programmers, democratizing access to complex reporting tasks. However, it could also deskill the workforce if users become overly dependent on the interface and lose the ability to diagnose underlying computational issues. The paper advocates for a hybrid model where the codeless system is used for routine generation, but training in the underlying programming concepts remains mandatory. This approach balances efficiency gains with professional development [20].

9. Conclusion

This paper has presented a workflow-oriented extension of the TLFQC platform that transforms it from a point solution for automated TLF generation into a comprehensive infrastructure for reproducible, versioned, and traceable clinical reporting. The architecture integrates specification versioning, containerized execution, immutable audit trails, and codeless validation into a cohesive system that addresses the systemic challenges of reproducibility in biostatistics. Key structural trade-offs, such as those between flexibility and auditability, have been examined, and governance policies for cross-study consistency have been proposed.

The implications of this work extend beyond individual organizations. As the pharmaceutical industry moves toward more data-driven and automated workflows, the need for robust reproducibility infrastructure becomes paramount. The FAIR principles, originally developed for data, can be extended to TLF outputs as digital objects that are findable through metadata, accessible through standardized APIs, interoperable across toolchains, and reusable through

provenance records. The extension of TLFQC serves as a concrete instantiation of these principles in a domain with high regulatory stakes.

Future research directions include the integration of natural language processing to automatically generate specification templates from statistical analysis plans, the use of machine learning to detect anomalies in output distributions across study updates, and the exploration of decentralized ledger technologies for cross-site audit trails in multi-center trials. By positioning reproducibility as a design requirement rather than an afterthought, the clinical reporting ecosystem can achieve higher levels of reliability, transparency, and trust.

References

1. Peng, R. D. (2011). Reproducible research in computational science. Science, 334(6060), 1226–1227.

2. U.S. Food and Drug Administration. (2003). Guidance for industry: Part 11, electronic records; electronic signatures — scope and application. https://www.fda.gov/regulatory-information/search-fda-guidance-documents/part-11-electronic-records-electronic-signatures-scope-and-application

3. U.S. Food and Drug Administration. (2017). Use of electronic health record data in clinical investigations: Guidance for industry. https://www.fda.gov/regulatory-information/search-fda-guidance-documents/use-electronic-health-record-data-clinical-investigations

4. Gentleman, R., & Temple Lang, D. (2007). Statistical analyses and reproducible research. Journal of Computational and Graphical Statistics, 16(1), 1–23.

5. Stodden, V., Leisch, F., & Peng, R. D. (Eds.). (2014). Implementing reproducible research. CRC Press.

6. Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., ... & Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data, 3(1), 160018.

7. Davidson, S. B., & Freire, J. (2008). Provenance and scientific workflows: Challenges and opportunities. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 1345–1350.

8. Boettiger, C. (2015). An introduction to Docker for reproducible research. ACM SIGOPS Operating Systems Review, 49(1), 71–79.

9. Koop, D., Santos, E., & Freire, J. (2008). Provenance-enabled data exploration and visualization with VisTrails. Proceedings of the 16th International Conference on Scientific and Statistical Database Management, 223–240.

10. Beaulieu-Jones, B. K., & Greene, C. S. (2017). Reproducibility of computational workflows is automated using continuous analysis. Nature Biotechnology, 35(4), 342–346.

11. Crowe, B. J., Xia, H. A., Berlin, J. A., Watson, D. J., Shi, H., Lin, S. L., ... & Wang, Y. (2009). Recommendations for safety planning, data collection, evaluation and reporting during drug, biologic and vaccine development: A report of the safety planning, evaluation, and reporting team. Clinical Trials, 6(5), 430–440.

12. Vaziri, M., Mandel, L., & Shinnar, A. (2018). Secure data provenance in a clinical trial setting. Journal of the American Medical Informatics Association, 25(6), 651–659.

13. Cito, J., Leitner, P., & Gall, H. C. (2015). The role of dependency management and reproducibility in software engineering. Proceedings of the 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, 412–415.

14. Holmes, S., & Huber, W. (2019). Modern statistics for modern biology. Cambridge University Press.

15. Patil, P., Peng, R. D., & Leek, J. T. (2016). A statistical definition for reproducibility and replicability. bioRxiv, 066803.

16. Hüllermeier, E., & Kruse, R. (2019). From machine learning to explainable AI. Informatik Spektrum, 42(5), 332–341.

17. Hendler, J. (2014). Data integration for heterogeneous biomedical datasets. Journal of Biomedical Informatics, 47, 1–3.

18. Ling, C., & Wang, Y. (2025). TLFQC: A High-compatible R Shiny based Platform for Automated and Codeless TLFs Generation and Validation. In PharmaSUG 2025 conference proceedings.

19. U.S. Food and Drug Administration. (2016). 21st Century Cures Act. Public Law 114-255.

20. Poldrack, R. A., & Poline, J. B. (2019). The publication and reproducibility challenges of neuroimaging. Nature Reviews Neuroscience, 20(8), 478–489.